

# PPP Packet Processor 622 Mbps MegaCore Function

---

PP622

August 2001  
User Guide



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
<http://www.altera.com>

A-UG-IPPP622-1.01

Copyright © 2001 Altera Corporation. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. All rights reserved.





## About this User Guide

### User Guide

This user guide provides comprehensive information about the Altera® PPP Packet Processor 622 Mbps MegaCore® Function (PP622).

Table 1 shows the user guide revision history.



Go to the following sources for more information:

- See “Features” on page 10 for a complete list of the core features, including new features in this release.
- Refer to the PP622 **readme** file for late-breaking information that is not available in this user guide.

**Table 1. User Guide Revision History**

Date	Description
December 2000	First version of user guide.
August 2001	First revision. Added “Core Verification Summary” section. Revised the “Getting Started” chapter.

## How to Find Information

- The Adobe Acrobat Find feature allows you to search the contents of a PDF file. Click on the binoculars icon in the top toolbar to open the Find dialog box, or click the right mouse button for a pull-down menu.
- Bookmarks serve as an additional table of contents.
- Thumbnail icons, which provide miniature previews of each page, provide a link to the pages.
- Numerous links, shown in green text, allow you to jump to related information.

## How to Contact Altera

For the most up-to-date information about Altera products, go to the Altera world-wide web site at <http://www.altera.com>.

For additional information about Altera products, consult the sources shown in [Table 2](#).

<b>Table 2. How to Contact Altera</b>			
<b>Information Type</b>	<b>Access</b>	<b>USA &amp; Canada</b>	<b>All Other Locations</b>
Altera Literature Services	Electronic mail	<a href="mailto:lit_req@altera.com">lit_req@altera.com</a> (1)	<a href="mailto:lit_req@altera.com">lit_req@altera.com</a> (1)
Non-technical customer service	Telephone hotline	(800) SOS-EPLD	(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)
	Fax	(408) 544-7606	(408) 544-7606
Technical support	Telephone hotline	(800) 800-EPLD (7:00 a.m. to 5:00 p.m. Pacific Time)	(408) 544-7000 (1) (7:30 a.m. to 5:30 p.m. Pacific Time)
	Fax	(408) 544-6401	(408) 544-6401 (1)
	Electronic mail	<a href="mailto:telecom@altera.com">telecom@altera.com</a>	<a href="mailto:telecom@altera.com">telecom@altera.com</a>
	FTP site	<a href="ftp.altera.com">ftp.altera.com</a>	<a href="ftp.altera.com">ftp.altera.com</a>
General product information	Telephone	(408) 544-7104	(408) 544-7104 (1)
	World-wide web site	<a href="http://www.altera.com">http://www.altera.com</a>	<a href="http://www.altera.com">http://www.altera.com</a>

**Note:**

(1) You can also contact your local Altera sales office or sales representative.

## Typographic Conventions

The *PPP Packet Processor 622 Mbps MegaCore Function (PP622) User Guide* uses the typographic conventions shown in [Table 3](#).

<i>Table 3. Conventions</i>	
Visual Cue	Meaning
<b>Bold Type with Initial Capital Letters</b>	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: <b>Save As</b> dialog box.
<b>bold type</b>	External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: <b>f<sub>MAX</sub></b> , <b>\maxplus2</b> directory, <b>d:</b> drive, <b>chiptrip.gdf</b> file.
<b><i>Bold italic type</i></b>	Book titles are shown in bold italic type with initial capital letters. Example: <b><i>1999 Device Data Book</i></b> .
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75 (High-Speed Board Design)</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t<sub>PIA</sub></i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pof file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of Quartus II and MAX+PLUS II Help topics are shown in quotation marks. Example: “Configuring a FLEX 10K or FLEX 8000 Device with the BitBlaster™ Download Cable.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix _n, e.g., reset_n.  Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\max2work\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier.
1., 2., 3., and a., b., c.,...	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
☞	The hand points to information that requires special attention.
↵	The angled arrow indicates you should press the Enter key.
👣	The feet direct you to more information on a particular topic.

## Abbreviations and Acronyms

ACCM	Asynchronous Control Character Map
AHDL	Altera Hardware Description Language
CPU	Central processing unit
CRC (16/32)	Cyclic Redundancy Check 16- or 32-bit
CRC-CCITT	Cyclic Redundancy Check
EDA	Electronic Design Automation
EDIF	Electronic Design Interchange Format
ESB	Embedded System Block
FCS	Frame Check Sequence
FIFO	First In First Out
HDL	Hardware Description Language
HDLC	High-Level Data Link Control
I/O	Input/Output
IP	Intellectual Property
LE	Logic Element
LSB	Least Significant Bit
LSByte	Least Significant Byte
Mbps	Megabits per second
MSB	Most Significant Bit
MSByte	Most Significant Byte
PC	Personal computer
POS-PHY	Packet Over SONET Physical layer
PPP	Point-to-Point Protocol
RX	Receive
RXHLDC	Receive High-Level Data Link Control sub-block
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Network
SPE	Synchronous Payload Envelope
STS-3	System Transport Signal level 3
TX	Transmit
TXHLDC	Transmit High-Level Data Link Control sub-block
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit



## About this User Guide

How to Find Information .....	iii
How to Contact Altera .....	iv
Typographic Conventions .....	v
Abbreviations and Acronyms .....	vi

## Specifications

General Description .....	9
Features .....	10
Receive Features .....	10
Transmit Features .....	10
Functional Description .....	11
Receiver Description .....	11
RXHDLC .....	11
Byte Alignment .....	12
Descrambling .....	13
Processing .....	13
FCS 16/32 .....	13
FCS Deletion .....	13
Transmitter Description .....	13
TXHDLC .....	13
Processing .....	14
FCS 16/32 .....	14
Scrambling .....	14
Interfaces & Protocols .....	14
Midbus Interface .....	14
AIRbus Interface .....	14
Atlantic Interface .....	15
Data Ordering .....	15
Transparency .....	15
I/O Signals .....	16
Performance .....	17
Software Interface .....	18
Memory Map .....	18
Registers .....	18
RX Register Description .....	19
RX_CTRL - Receive Control Register - 'h00 .....	19
RX_IS - Receive Interrupt Register - 'h02 .....	20
RX_IE - Receive Interrupt Enable Register - 'h04 .....	21

---

RX_PM_GOOD - Receive Good Packet Count - 'h06 .....	21
RX_PM_FCS - Receive FCS Error Count - 'h08.....	21
RX_PM_ABORT - Receive Abort Count - 'h0A .....	21
RX_PM_RUNT - Receive Runt Frame Count - 'h0C .....	22
TX Register Description .....	22
TX_CTRL - Transmit Control Register - 'h10 .....	22
TX_IS - Transmit Interrupt Register - 'h12 .....	23
TX_IE - Transmit Interrupt Enable Register - 'h14.....	23
TX_PM_GOOD - Transmit Good Packet Count - 'h16 .....	23
TX_PM_ERR - Transmit Error Packet Count - 'h18 .....	24
Core Verification Summary .....	24
Simulation Environment .....	24
Compatibility Testing Environment .....	25
<b>Getting Started</b> .....	
Design Walkthrough .....	27
Obtaining & Installing the PP622 .....	28
Downloading the MegaCore Function .....	28
Installing the MegaCore Files .....	28
Generating a PP622 .....	29
Implementing the System .....	30
Simulating Your Design .....	30
Using the Verilog Demo Testbench .....	30
Using the Visual IP Software .....	31
Synthesis, Compilation & Place & Route .....	31
Using Third-Party EDA Tools for Synthesis .....	31
Using the Quartus II development tool for compilation and place-and-route .....	31
Licensing for Configuration .....	32
Performing Post-Routing Simulation .....	32



## General Description


The PP622 encapsulates user data packets using HDLC-type framing, in compliance with the Internet Request For Comment—RFC 1622 and RFC 2615—documents. For a link to these documents, please visit the Internet Engineering Task Force web site at: <http://www.ietf.org/rfc>.

The following functions are not supported by the PP622:

- RFC1662 ACCM
- Bit stuffing
- Non-flag inter-frame fill
- Transparent mode (FCS not appended to transmit frame)
- Maximum length frame check

For the purpose of this user guide, "receive" indicates data flowing into the PP622 from the Midbus interface for transmission through the Atlantic™ interface; "transmit" indicates data received from the Atlantic interface for transmission through the Midbus interface. Thus the Atlantic interface is the source for transmit packets, and the sink for received packets.

The PPP Packet Processor 622 Mbps MegaCore® Function (PP622) uses the MegaWizard® Plug-In—within the Quartus® II software—to generate variants in VHDL, AHDL, or Verilog HDL, which you can instantiate into your design. Table 1 shows the optional features available to generate the PP622.

 Only the basic configuration of PP622 is available.

<b>Table 1. Optional Features</b> <i>Note (1)</i>				
<b>Options</b>	<b>Parameters</b>	<b>Choices</b>	<b>LEs</b>	<b>ESBs</b>
<b>Basic Configuration</b>	—	—	1,197	0

**Note:**

- (1) The numbers for the LEs and ESBs are approximate as of August 2001. Users are strongly advised to run the MegaWizard Plug-In and the Quartus II software to see exact numbers for the PP622.

## Features

### Receive Features

- Extraction of octet-synchronous PPP packets from a single STS-12 SPE
- Arbitrary packet length (one or more octets)
- Packet delineation and destuffing of all stuffed octets
- Byte-realignment (software programmable)
- Descrambling (software programmable)
- Error detection by CRC-32 and CRC-CCITT FCS
- Flag sequence deletion
- FCS deletion (software programmable)
- RFC1662 and RFC 2615 compliant HDLC-type framing (some sections are not implemented)
- Performance monitoring by counting:
  - Good frames
  - Bad FCS
  - Aborted frames
  - Runt frames
- Error detection:
  - FCS
  - Runt
- Optional removal of FCS

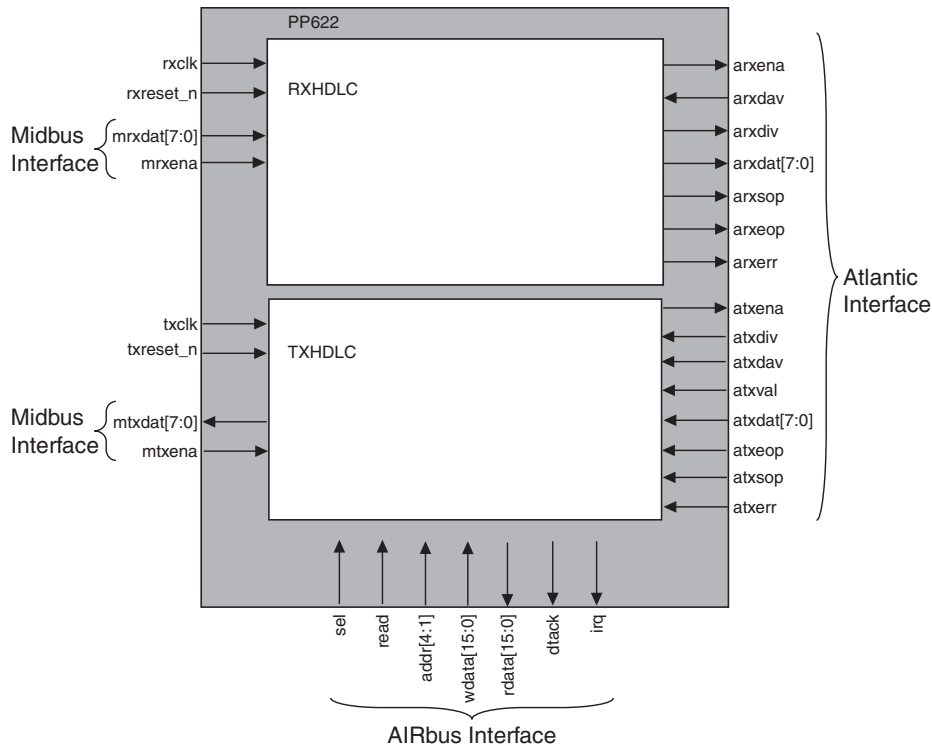
### Transmit Features

- RFC1662 and RFC 2615 compliant HDLC-type framing (some sections are not implemented)
- Flag octets and control-escape octets stuffing in message and FCS for transparency
- 16- or 32-bit FCS appended to packet
- Flag sequence insertion between packets for under utilized paths
- Flag sharing (single flag between frames)
- Scrambling (software programmable)
- Automatic abortion in case of transmit underflow, or by host command from Atlantic interface
- Arbitrary packet length (one or more octets)
- Automatic appending of FCS to transmitted frames
- Performance monitoring by counting:
  - Good frames
  - Underflows
  - Aborted packets

## Functional Description

The PP622 is a fully static, reusable, easy to replicate, single-channel processor (no interleaving) which operates in octet-synchronous mode, and offers full-duplex processing capability. [Figure 1](#) shows a complete block diagram of the PP622, including the three interfaces that support it. See “[Interfaces & Protocols](#)” on [page 14](#) for more information.

Figure 1. Block Diagram



## Receiver Description

The PP622 receives data from the Midbus interface and forwards it to the RXHDL block for processing.

### RXHDL

The RXHDL block processes the data in the incoming PPP packet. The following descriptions explain the principle functions of the RXHDL block.

*Byte Alignment*

Byte alignment aligns the boundaries of the PPP bytes (flags, CRC, etc.) with the boundaries of the packet processor. This function is necessary if the transmission facility does not preserve byte boundaries. If the byte boundaries are preserved (as in SONET), this function should be disabled.

Assume the following packet:

Clock Cycle	Contents	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	flag	0	1	1	1	1	1	1	0
2	flag	0	1	1	1	1	1	1	0
3	data	a	b	c	d	e	f	g	h
4	data	i	j	k	l	m	n	o	p
5	FCS	c	c	c	c	c	c	c	c
6	FCS	c	c	c	c	c	c	c	c
7	flag	0	1	1	1	1	1	1	0
8	flag	0	1	1	1	1	1	1	0

If the data becomes misaligned by 2 bits, the bytes on the Midbus are as follows:

Clock Cycle	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	x	x	0	1	1	1	1	1
2	1	0	0	1	1	1	1	1
3	1	0	a	b	c	d	e	f
4	g	h	i	j	k	l	m	n
5	o	p	c	c	c	c	c	c
6	c	c	c	c	c	c	c	c
7	c	c	0	1	1	1	1	1
8	1	0	0	1	1	1	1	1
9	1	0	x	x	x	x	x	x

Symptoms of misalignment are:

- Extremely long packets: They typically occur during line idle, when the bus is normally carrying flags. The flags no longer look like flags, but like packet data. In the above example, the flags are converted into 10011111.
- Excessive error rate: Random data patterns may, when shifted, masquerade as flags and hence mark the start and stop of packets. This is likely to cause FCS errors and, less likely, runts or aborts.

If it detects a misalignment, the aligner rearranges the data to compensate. This function is enabled via the Receive Control Register.

### *Descrambling*

The receive frame and between-frame flags are descrambled using a self-synchronizing descrambler. The generator polynomial used is the standard  $x^{43}+1$ . Descrambling can be enabled or disabled via the Receive Control Register. The inter-frame fill is then discarded, and the frame is forwarded for processing.

### *Processing*

Processing involves taking the descrambled frame, detecting the start of frame and end of frame indicated by flags, and removing the stuff octets (control escape characters). Aborted frames are also detected.

### *FCS 16/32*

The FCS 16/32 blocks check for errors by calculating a syndrome, and by using either the CRC-16 ( $x^{16} + x^{12} + x^5 + 1$ ) or CRC-32 ( $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ ) generator polynomial as an FCS. All packets—good packets, packets with CRC errors, aborted packets, and runt packets—are counted in statistics registers.

### *FCS Deletion*

FCS Deletion removes the FCS from the frame. Removal of the FCS is enabled or disabled via the Receive Control Register. The frame is then passed to the Atlantic interface.

The PP622 takes in packets from the Atlantic interface and passes them to the TXHDLC block for processing.

## **TXHDLC**

The TXHDLC block frames the data in the incoming transmit stream. The following descriptions explain the principle functions of the TXHDLC block.

## **Transmitter Description**

### *Processing*

The TXHDLC block inserts the FCS into the data stream for stuffing at the end of a normal packet. The TXHDLC also inserts the abort sequence as needed into the data stream for aborted packets, and inserts flags between frames. It also stuffs the data and FCS octets for transparency. Stuffing is performed 8 bits at a time (byte oriented) for high performance, and low gate count. When enabled, it delivers octets to the Midbus interface.

### *FCS 16/32*

The FCS is calculated using the CRC-16 or CRC-32 generating polynomial.

### *Scrambling*

The transmit data stream is scrambled using the  $x^{43}+1$  polynomial. Scrambling can be enabled or disabled via the Transmit Control Register. The packet is sent to the Midbus interface for transmission.

## Interfaces & Protocols

### **Midbus Interface**

The Midbus interface is a simple synchronous full-duplex data path bus. The PP622 Midbus runs at 77.76 MHz over a single byte lane in each direction. In the receive direction (RX), data is transferred from the Midbus master to the slave (PP622). In the transmit direction (TX), data is transferred from the slave (PP622) to the master. In each direction, the Midbus can carry 8 bits per clock cycle. It includes Midbus receive data (`mrxd[7:0]`) and Midbus receive enable (`mrxena`) lines to indicate valid data transfers in the RX direction, and Midbus transmit data (`mtxd[7:0]`) and Midbus transmit enable (`mtxena`) lines to indicate valid data requests in the TX direction. Since the PP622 is a slave to the Midbus it can work with any Midbus master.

### **AIRbus Interface**

The AIRbus interface provides access to internal registers using a simple synchronous internal bus protocol. This consists of separate read data (`rdata[15:0]`) and write data (`wdata[15:0]`) buses, a data transfer acknowledge (`dtack`) signal, and a select (`sel`) signal. An address (`addr[4:1]`) bus and read (`read`) signal indicate the location and type of access within the block. The `rdata` buses and `dtack` signals can be merged from multiple blocks using a simple OR function. The `dtack` signal is sustained until the block `sel` is removed (four-way handshaking) meaning the AIRbus can cross clock domain boundaries. The PP622 is an AIRbus slave with a data width of 16 bits.

## Atlantic Interface

The Atlantic interface is a full-duplex synchronous bus protocol supporting both packets and cells. The PP622 is an Atlantic interface master using an 8-bit wide data path, clocked at 77.76 MHz, to deliver packets to the slave.



The `arxdav`, `atxdiv`, and `atxsop` signals are provided, but are not used.



More detailed information on the Midbus, AIRbus, and Atlantic is available from the Altera web site at <http://www.altera.com>.

### Data Ordering

Data is transferred in parallel (8 bits at a time) in the same order it was received internally between the PP622 and the interfaces, but it is ultimately transmitted/received MSB first through external bit-serial mediums such as SONET and SDH (optical fibre), in keeping with telephony convention.

This convention of MSB first is not followed in doing CRC (FCS/syndrome) calculations, which use data communications LSB first conventions. In CRC calculations, the LSB of each octet is nominally processed first. CRC calculations are actually done in parallel though the calculation simulates a bit-serial LSB first calculation. The order that octets go into the CRC calculation is the same as the transmission order. The difference in ordering has no practical effect as long as both ends follow the same convention.



It is important to specify the nominal order within the bits on the Atlantic and Midbus interfaces.

### Transparency

Special flag characters delimit the start and end of a PPP frame. These special characters are known as a Flag Sequence, which is the binary sequence 01111110 (hexadecimal 0x7e). There is no length field, loss of carrier, or other delimiter of packet boundaries. Since the data in the packet is unrestricted, all data patterns are valid. Transparency is achieved when the framing protocol stuffs flag and escape characters as explained in RFC1662:

“An octet stuffing procedure is used. The Control Escape octet is defined as binary 01111101 (hexadecimal 0x7d), most significant bit first (...)

After FCS computation, the transmitter examines the entire frame between the two Flag Sequences. Each Flag Sequence, Control Escape octet (...) is replaced by a two octet sequence consisting of the Control Escape octet followed by the original octet exclusive-or'd with hexadecimal 0x20 (...)

0x7e is encoded as 0x7d, 0x5e. (Flag Sequence)  
0x7d is encoded as 0x7d, 0x5d. (Control Escape) (...)"

RFC1662 also defines an ACCM for the convenience of software and hardware designed for transferring text files. This allows other special characters to be stuffed as directed by the user. The PP155 does not implement the ACCM, thus the TXHDLIC does not stuff anything other than the flag and control escape patterns. However, in accordance with RFC1662, the RXHDLIC destuffs any stuffed octets.

"On reception (...) Each Control Escape octet is also removed, and the following octet is exclusive-or'd with hexadecimal 0x20, unless it is the Flag Sequence (which aborts a frame).

## I/O Signals

Table 2 describes the input/output signals used by the PP622.

<b>Table 2. I/O Signals (Part 1 of 2)</b>		
<b>Port</b>	<b>Direction</b>	<b>Description</b>
<b>Receive Interface Signals</b>		
rxclk	Input	Clock
rxreset_n	Input	Active low reset
<b>Midbus Receive Interface Signals</b>		
mrxena	Input	Enable
mrxd[7:0]	Input	Data bus
<b>Atlantic Receive Interface Signals</b>		
arxena	Output	Enable
arxdav	Input	Data available
arxdiv	Output	Data divider
arxd[7:0]	Output	Data bus
arxsop	Output	Start of packet
arxeop	Output	End of packet
arxerr	Output	Error indication



<b>Table 2. I/O Signals (Part 2 of 2)</b>		
<b>Port</b>	<b>Direction</b>	<b>Description</b>
<b>Transmit Interface Signals</b>		
txclk	Input	Clock
txreset_n	Input	Active low reset
<b>Midbus Transmit Interface Signals</b>		
mtxena	Input	Enable
mtxdat[7:0]	Output	Data bus
<b>Atlantic Transmit Interface Signals</b>		
atxena	Output	Enable
atxdav	Input	Data available
atxdiv	Input	Data divider
atxval	Input	Enable
atxdat[7:0]	Input	Data bus
atxsop	Input	Start of packet
atxeop	Input	End of packet
atxerr	Input	Error indication
<b>AIRbus Interface Signals</b>		
sel	Input	Select
read	Input	Read: High for read cycles, Low for writes
addr[4:1]	Input	Address
wdata[15:0]	Input	Write data
rdata[15:0]	Output	Read data: All zeros if sel is not asserted.
dtack	Output	Data transfer acknowledge
irq	Output	Interrupt request

## Performance

Table 3 shows the required speed and estimated gate count of PP622 in an APEX 20KE device.

<b>Table 3. Performance</b> <i>Note (1)</i>		
<b>LEs</b>	<b>ESBs</b>	<b>f<sub>MAX</sub> (MHz)</b>
1,197	0	77.76 required to support 622.08 Mbps

**Note:**

(1) The numbers for the LEs and ESBs are approximate as of August 2001.

## Software Interface

### Memory Map

All addresses access 16-bit registers and are shown as hexadecimal values. The value is the byte address, thus bit 0 is not used. All addresses are even.

Address	Register	Description
'h00	RX_CTRL	Receive Control Register
'h02	RX_IS	Receive Interrupt Register
'h04	RX_IE	Receive Interrupt Enable Register
'h06	RX_PM_GOOD	Receive Good Packet Count
'h08	RX_PM_FCS	Receive FCS Error Count
'h0A	RX_PM_ABORT	Receive Abort Count
'h0C	RX_PM_RUNT	Receive Runt Frame Count
'h10	TX_CTRL	Transmit Control Register
'h12	TX_IS	Transmit Interrupt Register
'h14	TX_IE	Transmit Interrupt Enable Register
'h16	TX_PM_GOOD	Transmit Good Packet Count
'h18	TX_PM_ERR	Transmit Error Packet Count

### Registers

The following is a list of access codes used to describe the type of register bits.

Code	Description
RW	Read/Write
RO	Read-Only
RW1C	Read/Write 1 to Clear
RW0S	Read/Write 0 to Set
RTC	Read to Clear
RTS	Read to Set
RTCW	Read to Clear/Write
RTSW	Read to Set/Write
RWTC	Read/Write any value to Clear
RWTS	Read/Write any value to Set
RWSC	Read/Write Self-Clearing
RWSS	Read/Write Self-Setting
UR0	Unused bits/Read as 0
UR1	Unused bits/Read as 1

*RX Register Description*

The following tables describe the registers for the receiver section of the PP622.

<b><i>RX_CTRL - Receive Control Register - 'h00 (Part 1 of 2)</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
REALIGN	5	RWSC	1: Forces aligner to try new alignment  Writing a 1 to this bit (note that it always reads as 0) forces the byte aligner to try a new alignment. This bit is ignored if align is 0.	0
ALIGN	4	RW	0: Byte alignment disabled. 1: Byte alignment enabled.  If set, the aligner monitors the error rate to determine if the PPP bytes are aligned to mrxdad. If data are not aligned, it shifts the data on mrxdad to attempt to align the data. If cleared, the PPP bytes are assumed to be aligned to mrxdad.	0
CRCLEN	3	RW	0: CRC-CCITT mode 1: CRC-32 mode  If set, the receiver computes syndromes for received packets using the CRC-32 polynomial, if cleared, the CRC-CCITT 16-bit polynomial. The syndrome is always computed and checked. Packets with bad syndromes (i.e. with errors) are not discarded, but are counted in the Receive FCS Error Count register and marked as erroneous when sent to the Atlantic interface.	0
SCRAMEN	2	RW	Enable descrambling of transmit frame and descrambling of the receive frame.  This causes the received data stream to be descrambled using the $x^{43}+1$ scrambling polynomial. The bits are descrambled using the most significant (higher-numbered) bit first. All bits, including flags, the body of the packet, and the FCS are scrambled. The descrambling process is self-synchronizing. The descrambler is not reset by the enable bit and will therefore synchronize the state with the received data stream while the enable is cleared. It should, therefore, be set to its desired value and at least 43 bits of data should be received before turning on the enable bit.	0

<b><i>RX_CTRL - Receive Control Register - 'h00 (Part 2 of 2)</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
DELFCS	1	RW	<p>0: Store FCS with received packet. 1: Delete FCS from received frame.</p> <p>If cleared, the packet is delivered to the Atlantic interface with the FCS appended. Setting this bit causes the receiver to remove the FCS before sending the packet to the Atlantic interface. Users may set this bit to improve the performance of the Atlantic interface or to simplify downstream processing of the packet. This function requires that the crclen bit be set properly.</p>	0
ENABLE	0	RW	<p>0: Disable packet reception. 1: Enable packet reception.</p> <p>Clearing this bit to zero places the internal state variables in the idle condition, clears the status register, and all receive performance monitor counts (good, FCS, runt, abort). Setting it enables all functions.</p>	0

<b><i>RX_IS - Receive Interrupt Register - 'h02</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
GOOD	4	RW1C	<p>Set if a good packet is received.</p> <p>This is set if a legal, correct packet is received.</p>	0
UNALIGNED	3	RW1C	<p>Set if receiver is unaligned.</p> <p>This is set when the receive aligner is not in the aligned state. To determine when the aligner has regained alignment, software should read and clear this bit periodically until it stays cleared. Clearing align in the control register prevents this bit from being set, but does not clear the bit.</p>	0
FCS	2	RW1C	<p>Set if an FCS error occurs.</p> <p>This is set if a packet with an incorrect FCS is received.</p>	0
ABORT	1	RW1C	<p>Set if a receive packet is aborted.</p> <p>This is set if a packet terminated with an abort sequence is received.</p>	0
RUNT	0	RW1C	<p>Set if a runt packet is received.</p> <p>This is set if an illegally short packet is received.</p>	0

***RX\_IE - Receive Interrupt Enable Register - 'h04***

Field	Bits	Access	Function	Default
GOOD	4	RW	Enables the receive good interrupt status.	0
UNALIGNED	3	RW	Enables the receive unaligned interrupt status.	0
FCS	2	RW	Enables the receive FCS interrupt status.	0
ABORT	1	RW	Enables the receive abort interrupt status.	0
RUNT	0	RW	Enables the receive runt interrupt status.	0

***RX\_PM\_GOOD - Receive Good Packet Count - 'h06***

Field	Bits	Access	Function	Default
CNT	15:0	RTCW	Count of correctly received packets.  Each correctly received packet (good FCS, not aborted, not a runt) causes this count to increment by one. The counter saturates at 0xFFFF: it will not wrap to 0. Reading this register clears the count. 16-bit values may be written to this register for testing.	0

***RX\_PM\_FCS - Receive FCS Error Count - 'h08***

Field	Bits	Access	Function	Default
CNT	15:0	RTCW	Count of received packets with bad FCS.  Packets that are not runts and are not aborted but which have bad CRC syndromes increment this count. The counter saturates at 0xFFFF and does not wrap to 0. Reading this register clears the count.	0

***RX\_PM\_ABORT - Receive Abort Count - 'h0A***

Field	Bits	Access	Function	Default
CNT	15:0	RTCW	Count of received packets terminated with ABORT.	0

<b><i>RX_PM_RUNT - Receive Runt Frame Count - 'h0C</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
CNT	15:0	RTCW	Count of runt frames.  Runt packets are packets received with frames shorter than 3 or 5 octets (depending on the setting of crclen), not including stuff octets and flags. Individual runt packets increment the count. The counter saturates at 0xFFFF and does not wrap to 0. Reading this register clears the count. 32-bit values may be written to this register for testing.	0

### *TX Register Description*

The following tables describe the registers for the transmitter section of the PP622.

<b><i>TX_CTRL - Transmit Control Register - 'h10</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
CRCLEN	2	RW	Set to 1 for CRC-32 (32 bits); 0 for CRC-CCITT (16 bits).  If set, the transmit section computes FCSs for transmit packets using the CRC-32 polynomial; if cleared, the CRC-CCITT 16-bit polynomial is used. The FCS is always generated and appended.	0
SCRAMEN	1	RW	Enables scrambling of the transmit frame.  This causes the transmitted data stream to be scrambled using the $x^{43}+1$ scrambling polynomial. The bits are scrambled using the most significant (higher-numbered) bit first. All bits, including flags, the body of the packet, and the FCS are scrambled.	0
ENABLE	0	RW	0: Disable packet transmission. 1: Enable packet transmission.  Clearing this bit to zero places the internal state variables in the idle condition, clears the status registers and all transmit performance monitor counts (good, error). Setting it enables all functions.	0

<b><i>TX_IS - Transmit Interrupt Register - 'h12</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
GOOD	2	RW1C	Set if a packet is transmitted correctly.  This bit is set by the transmission of a complete packet with no errors.	0
ERR	1	RW1C	Set if a packet is aborted (atxerr asserted simultaneously with atxeop).  This is set if atxerr is asserted. Software writes a 1 to this bit to clear it.	0
UNDERFLOW	0	RW1C	Set if the FIFO has underflowed.  This bit is set if the Atlantic interface does not supply data when required (i.e atxval = 0 in the middle of a packet). If atxval is negated after sending a complete packet and before starting a new packet no underflow is reported. An underflow aborts the current packet and discards the remainder. Subsequent packets are transmitted normally. Software writes a 1 to this bit to clear it.	0

<b><i>TX_IE - Transmit Interrupt Enable Register - 'h14</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
GOOD	2	RW	Enables the transmit good interrupt status.	0
ERR	1	RW	Enables the transmit err interrupt status.	0
UNDERFLOW	0	RW	Enables the transmit underflow interrupt status.	0

<b><i>TX_PM_GOOD - Transmit Good Packet Count - 'h16</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
CNT	15:0	RTCW	Count of correctly transmitted packets.  Each correctly transmitted packet (no underflow, not aborted) causes this count to increment by one. The counter saturates at 0xFFFF; it does not wrap to 0. Reading this register clears the count. 16-bit values may be written to this register for testing.	0

<b><i>TX_PM_ERR - Transmit Error Packet Count - 'h18</i></b>				
<b>Field</b>	<b>Bits</b>	<b>Access</b>	<b>Function</b>	<b>Default</b>
CNT	15:0	RTCW	Count of transmitted packets terminated with ABORT.  Packets which are not runts but which are terminated by an abort sequence increment this count. The counter saturates at 0xFFFF; it does not wrap to 0. Reading this register clears the count.	0

## Core Verification Summary

The PP622 was the object of very thorough verification for operation according to industry standards. The PP622 was tested by simulation, and in-circuit for third-party compatibility. Both test environments are described briefly, including the number of test programs, and their results.

### Simulation Environment

The PP622 was simulated using behavioral utilities with multiple simulators, including but not limited to ModelSim SE. The behavioral utilities consist of generic flow control generators and monitors, Midbus generators and monitors, Atlantic generators and monitors, an AIRbus master model, and clock generators.

A test suite using the utilities and the RTL model of the PP622 was used to verify the proper operation of all the features listed on [page 10](#).

[Table 4](#) lists the results of the simulation for the PP622

<b><i>Table 4. Results</i></b>	
Number of test programs <a href="#">(1)</a>	64
Number of test programs passing	64
Number of test programs failing	0
Number of test cases <a href="#">(1)</a>	118
Number of test cases passing	118
Number of test cases failing	0

**Note:**

- (1) Each test program contains at least one test case.



Compatibility Testing Environment

The PP622 was evaluated—within an APEX EP20K1000EFC672 device—against a commercial third-party packet processor with similar features, as required by industry standards. For testing purposes, the PP622 was interconnected with the Altera STS12CFRM MegaCore function, and the Altera PCI MegaCore function was used to interface to the AIRbus and to the third-party packet processor. Figure 2 shows the test board used.

Software from a host PC was used to set registers on the PP622, and the third-party packet processor. The effects of setting these registers, and any corresponding registers, were observed to determine functionality. Tests were run for extended periods of time, thereby testing millions of packets.

Figure 2. Test Board

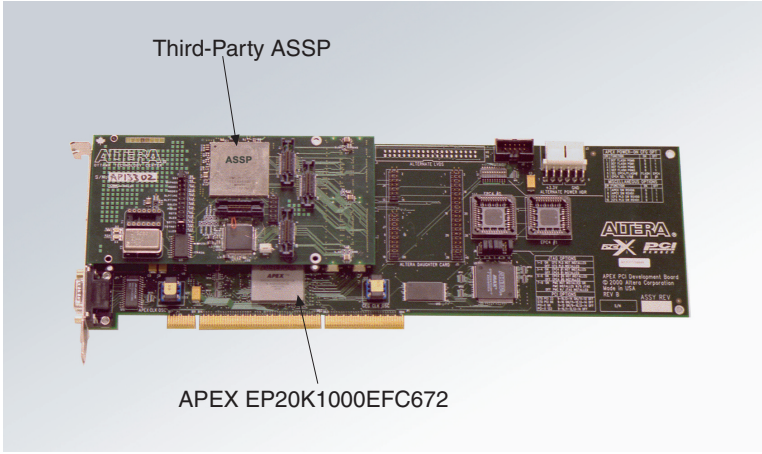


Table 5 lists the results of the hardware verification for the PP622.

Table 5. Results	
Number of test programs (1)	24
Number of test programs passing	24
Number of test programs failing	0
Number of test cases (1)	27
Number of test cases passing	27
Number of test cases failing	0

Note:

(1) Each test program contains at least one test case.



*Notes:*



This section describes how to obtain a variant from the PPP Packet Processor 622 Mbps MegaCore® Function (PP622). It explains how to install the PP622 on your PC, and walks you through the process of implementing the variant in a design.

You can test-drive a PP622 using the Altera® OpenCore® feature—within the Quartus® II software—to instantiate it, to perform place-and-route, to perform static timing analysis, and to simulate it using a third-party simulator, within your custom logic. You only need licenses when you are ready to generate programming files.

## Design Walkthrough

This design walkthrough involves the following steps:

1. Obtaining and installing the PP622 MegaCore Function.
2. Generating a PP622 for your system using the MegaWizard® Plug-In.
3. Implementing the rest of your system using AHDL, VHDL, or Verilog HDL.
4. Simulating the PP622 within your design.
5. Synthesis, compilation, and place-and-route.
6. Licensing the PP622 to configure the device.
7. Performing post-routing simulation.

The instructions assume that:

- You are using a PC
- You are familiar with the Quartus II software.
- The Quartus II software (the newest version) is installed in the default location.
- You are using the OpenCore feature to test-drive a PP622, or you have licensed it.

## Obtaining & Installing the PP622

To start using the PP622, you need to obtain the MegaCore package, which includes the following:

- Data sheet
- User guide
- AIRbus, Midbus, and Atlantic interface functional specifications
- MegaWizard Plug-In
  - Encrypted gate level netlist
  - Place-and-route constraints (where necessary)
  - Secure RTL simulation model
- Demo testbench
- Access to problem reporting system

### Downloading the MegaCore Function

If you have Internet access, you can download the PPP Packet Processor 622 Mbps MegaCore function from the Altera web site. Follow the instructions below to obtain the core via the Internet. If you do not have Internet access, you can obtain the core from your local Altera representative.

1. Point your web browser at <http://www.altera.com/IPmegastore>.
2. In the IP MegaSearch **keyword** field type PPP.
3. Click the link for the PPP Packet Processor 622 Mbps MegaCore function.
4. On the product page, click the *Free Test-Drive* icon.
5. Follow the on-line instructions to download the function and save it to your hard disk.

### Installing the MegaCore Files

Use the MegaWizard Plug-In to generate the files and install them on your PC. The following instructions describe this process.

For UNIX systems, you must have Java runtime environment version 1.3 before you can use the MegaWizard Plug-In. You can download this file from the Java web site at <http://www.java.sun.com>.

For Windows, follow the instructions below:

1. Click **Run** (Start menu).

2. Type `<path name>\<filename>.exe`, where `<path name>` is the location of the downloaded PP622 and `<filename>` is the filename of the PP622. Click **OK**.
3. The **MegaCore Installer** dialog box appears. Follow the MegaWizard Plug-In instructions to finish the installation.
4. Disregard this step if you are using Quartus II version 1.1 or higher. Otherwise, after you have finished installing the files, you must specify the directory in which you installed them as a user library in the Quartus II software. Search for “User Libraries” in Quartus II Help for instructions on how to add these libraries.

## Generating a PP622

This section describes the design flow using the PPP Packet Processor 622 Mbps MegaCore function and the Quartus II development system. A MegaWizard Plug-In is provided with the PP622. The MegaWizard Plug-In Manager—used within the Quartus II software—allows you to create or modify design files to meet the needs of your application. You can then instantiate the PP622 in your design file.

To create a PP622 using the MegaWizard Plug-In, follow these steps:

1. Start the MegaWizard Plug-In by choosing the MegaWizard Plug-In Manager command (Tools menu) in the Quartus II software. The **MegaWizard Plug-In Manager** dialog box is displayed.



Refer to Quartus II Help for detailed instructions on how to use the MegaWizard Plug-In Manager.

2. Specify that you want to create a new custom variant and click **Next**.
3. On the second page of the MegaWizard Plug-In, open the **Communications** folder, and select the PP622 from the PPP folder.
4. Choose the type of output files (language), specify the folder and name for the files the MegaWizard Plug-In creates, and click **Next**.
5. The final screen lists the design files created by the MegaWizard Plug-In, and indicates the location of the simulation models for the PP622. Click **Finish**.

## Implementing the System

Once you have created your PP622, you are ready to implement it. You can use the files generated by the MegaWizard Plug-In, and use the Quartus II software or other EDA tools to create your design. [Table 1](#) lists the generated files.

<b>Table 1. MegaWizard Plug-In Files</b>			
<b>Description</b>	<b>Verilog HDL</b>	<b>VHDL</b>	<b>AHDL</b>
Design File Wrapper	<b>.v</b>	<b>.vhd</b>	<b>.tdf</b>
Sample Instantiation	<b>_inst.v</b>	<b>_inst.vhd</b>	<b>_inst.tdf</b>
Black Box Module	<b>_bb.v</b>	–	–
Symbol files for the Quartus II software used to instantiate the PP622 into a schematic design	<b>.bsf</b>	<b>.bsf</b>	<b>.bsf</b>
An encrypted HDL netlist file	<b>.e.vqm.v</b>	<b>.e.vqm.v</b>	<b>.e.vqm.v</b>

## Simulating Your Design

Altera provides three models to be used for functional verification of the PP622 within your design. A Verilog demo testbench, including scripts to run it, is also provided. This demo testbench used with the ModelSim AE simulator demonstrates how to instantiate a model in a design.

To find the simulation models for your selected variant, refer to the last page of the MegaWizard Plug-In Manager. These models and the demo testbench are located on your hard drive, the paths are:

- `sim_lib/<variant>/modelsim_verilog/`
- `sim_lib/<variant>/modelsim_vhdl/`
- `sim_lib/<variant>/visual_ip/`
- `sim_lib/<variant>/test/`



`<variant>` is a unique code (aotXXXX\_#\_pp622) assigned to the specific configuration requested through the MegaWizard Plug-In.

### Using the Verilog Demo Testbench

The demo testbench includes some simple stimulus to control the user interfaces of the PP622. Each PP622 variant includes scripts to compile and run the demo testbench using a variety of simulators and models.

## Synthesis, Compilation & Place & Route

### Using the Visual IP Software

The Visual IP software facilitates the use of Visual IP simulation models with third-party simulation tools. To view a simulation model, you must have the Visual IP software installed on your system. To download the software, or for instructions on how to use the software, refer to the Altera web site at <http://www.altera.com>, and search for Visual IP. For examples of how to use the provided Visual IP model, refer to the sample scripts included with the demo testbench.

After you have verified that your design is functionally correct, you are ready to perform synthesis and place-and-route. Synthesis can be performed by the Quartus II development tool, or by a third-party synthesis tool. The Quartus II software works seamlessly with tools from many EDA vendors, including Cadence, Exemplar Logic, Mentor Graphics, Synopsys, Synplicity, and Viewlogic.

### Using Third-Party EDA Tools for Synthesis

To synthesize your design in a third-party EDA tool, follow these steps:

1. Create your custom design instantiating a PP622.
2. Synthesize the design using your third-party EDA tool. Your EDA tool should treat the PP622 instantiation as a black box by either setting attributes or ignoring the instantiation.
3. After compilation, generate a netlist file in your third-party EDA tool.

### Using the Quartus II development tool for compilation and place-and-route

To use the Quartus II software to compile and place-and-route your design, follow these steps:

1. Select **Compile mode** (Processing menu).
2. Specify the Compiler settings in the **Compiler Settings** dialog box (Processing menu) or use the Compiler Settings wizard.
3. Disregard this step if you are using Quartus II version 1.1 or higher. Otherwise, specify the user libraries for the project and the order in which the Compiler searches the libraries.

## 2

### Getting Started

4. Specify the input settings for the project. Choose **EDA Tool Settings** (Project menu). Select **Custom EDIF** in the Design entry/synthesis tool list. Click **Settings**. In the **EDA Tool Input Settings** dialog box, make sure that the relevant tool name or option is selected in the **Design Entry/Synthesis Tool** list.
5. Add your third-party EDA tool-generated netlist file to your project.
6. Add any **.tdf**, **.vhd**, or **.v** files not synthesized in the third-party tool.
7. Add the pre-synthesized and encrypted **.e.vqm.v** file from your working directory, created by the MegaWizard Plug-In Manager.
8. Constrain your design as needed.
9. Compile your design. The Quartus II Compiler synthesizes and performs place-and-route on your design.

Refer to Quartus II Help for further instructions on performing compilation.

## Licensing for Configuration

After you have compiled and analyzed your design, you are ready to configure your targeted Altera semiconductor device. If you are evaluating the PP622 with the OpenCore feature, you must license the function before you can generate programming files. To obtain licenses contact your local Altera sales representative.



All current PP622 variants use a single license, with ordering code: PLSM-PP622.

## Performing Post-Routing Simulation

After you have licensed the PP622, you can generate EDIF, VHDL, Verilog HDL, and Standard Delay Output Files from the Quartus II software and use them with your existing EDA tools to perform functional modeling and post-routing simulation of your design.

1. Open your existing Quartus II project.
2. Depending on the type of output file you want, specify Verilog HDL output settings or VHDL output settings in the **General Settings** dialog box (Project menu).
3. Compile your design with the Quartus II software, refer to the [“Using the Quartus II development tool for compilation and place-and-route”](#) section. The Quartus II software generates output and programming files.



4. You can now import your Quartus II software-generated output files (**.edo**, **.vho**, **.vo**, or **.sdo**) into your third-party EDA tool for post-route, device-level, and system-level simulation.



*Notes:*